Favourite number

Everyone has a favourite number. Jacob's favourite number is X and Jayden's favourite number is Y. A non-empty zero-indexed array A consisting of N integers is given. Jacob and Jayden are interested in occurrences of their favourite numbers X and Y in array A. They are looking for the longest leading fragment (*prefix*) of array A in which there is an equal number of occurrences of X and Y. More formally, they are looking for the largest P, such that 0 <= P < N and the number of occurrences of X equals the number of occurrences of Y in the sequence A[0], A[1], …., A[P].

For example, consider X = 7, Y = 42 and the following array A:

A[0] = 6
A[1] = 42
A[2] = 11
A[3] = 7
A[4] = 1
A[5] = 42

There are three prefixes of array A containing the same number of occurrences of X and Y:

- P = 0: A[0..0] = [6] contains neither 7 or 42
- P = 3: A[0..3] = [6, 42, 11, 7] contains one 7 and one 42
- P = 4: A[0..4] = [6, 42, 11, 7, 1] contains one 7 and one 42.

The largest value of P we are looking for is 4, because the only longer corresponding prefix A[0..5] contains one 7 and two 42's.

Jacob and Jayden have implemented a function:

```
object Solution { def solution(x: Int, y: Int, a: Array[Int]): Int }
```

which, given integers X, Y and a non-empty zero-indexed array A consisting of N integers, returns the maximum value of P for which A[0..P] contains the same number of occurrences of X and Y, or -1 if no such value exists.

For example, given integers X, Y and array A as defined above, the function should return 4, as explained above.

Assume that:

- N is an integer within the range [1..100,000]
- X and Y are integers within the range [1..1.000,000,000]
- each element of array A is an integer within the range [1..1,000,000,000].

Complexity:

- expected worst-case time complexity is O(N)
- expected worst-case space complexity is O(1), beyond input storage (not counting the storage required for input arguments).

Elements of input array can be modified.